Contents

Mirroring

4 minute read   ✓ page test

This task demonstrates the traffic mirroring capabilities of Istio.

Traffic mirroring, also called shadowing, is a powerful concept that allows feature teams to bring changes to production with as little risk as possible. Mirroring sends a copy of live traffic to a mirrored service. The mirrored traffic happens out of band of the critical request path for the primary service.

In this task, you will first force all traffic to `v1` of a test service. Then, you will apply a rule to mirror a portion of traffic to `v2`.

Before you begin

- Set up Istio by following the instructions in the Installation guide.
- Start by deploying two versions of the httpbin service that have access logging enabled:

  httpbin-v1:

  ```
  $ cat <<EOF | istioctl kube-inject -f - | kubectl create -f -
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: httpbin-v1
  spec:
    replicas: 1
    selector:
      matchLabels:
        app: httpbin
        version: v1
    template:
  ```

```
    metadata:
      labels:
        app: httpbin
        version: v1
    spec:
      containers:
      - image: docker.io/kennethreitz/httpbin
        imagePullPolicy: IfNotPresent
        name: httpbin
        command: ["gunicorn", "--access-logfile", "-", "-b", "0.0.0.0:
80", "httpbin:app"]
        ports:
        - containerPort: 80
EOF
```

httpbin-v2:

```
$ cat <<EOF | istioctl kube-inject -f - | kubectl create -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpbin-v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
      version: v2
  template:
    metadata:
      labels:
        app: httpbin
        version: v2
    spec:
      containers:
      - image: docker.io/kennethreitz/httpbin
        imagePullPolicy: IfNotPresent
        name: httpbin
        command: ["gunicorn", "--access-logfile", "-", "-b", "0.0.0.0:
80", "httpbin:app"]
        ports:
        - containerPort: 80
EOF
```

httpbin Kubernetes service:

```
$ kubectl create -f - <<EOF
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
spec:
  ports:
  - name: http
    port: 8000
    targetPort: 80
  selector:
    app: httpbin
EOF
```

- Start the `sleep` service so you can use `curl` to provide load:

  sleep service:

```
$ cat <<EOF | istioctl kube-inject -f - | kubectl create -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sleep
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sleep
  template:
    metadata:
      labels:
        app: sleep
    spec:
      containers:
      - name: sleep
        image: curlimages/curl
        command: ["/bin/sleep","3650d"]
        imagePullPolicy: IfNotPresent
EOF
```

Creating a default routing policy

By default Kubernetes load balances across both versions of the `httpbin` service. In this step, you will change that behavior so that all traffic goes to `v1`.

1. Create a default route rule to route all traffic to `v1` of the service:

    ```
    $ kubectl apply -f - <<EOF
    apiVersion: networking.istio.io/v1alpha3
    kind: VirtualService
    metadata:
      name: httpbin
    spec:
    ```

```
  hosts:
    - httpbin
  http:
  - route:
    - destination:
        host: httpbin
        subset: v1
      weight: 100
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: httpbin
spec:
  host: httpbin
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
```

```
EOF
```

Now all traffic goes to the `httpbin:v1` service.

2. Send some traffic to the service:

```
$ export SLEEP_POD=$(kubectl get pod -l app=sleep -o jsonpath={.items.
.metadata.name})
$ kubectl exec "${SLEEP_POD}" -c sleep -- curl -sS http://httpbin:8000
/headers
{
  "headers": {
    "Accept": "*/*",
    "Content-Length": "0",
    "Host": "httpbin:8000",
    "User-Agent": "curl/7.35.0",
    "X-B3-Parentspanid": "57784f8bff90ae0b",
    "X-B3-Sampled": "1",
    "X-B3-Spanid": "3289ae7257c3f159",
    "X-B3-Traceid": "b56eebd279a76f0b57784f8bff90ae0b",
    "X-Envoy-Attempt-Count": "1",
    "X-Forwarded-Client-Cert": "By=spiffe://cluster.local/ns/default/s
a/default;Hash=20afebed6da091c850264cc751b8c9306abac02993f80bdb7628223
7422bd098;Subject=\"\";URI=spiffe://cluster.local/ns/default/sa/defaul
t"
  }
}
```

3. Check the logs for `v1` and `v2` of the `httpbin` pods. You should see access log entries for `v1` and none for `v2`:

```
$ export V1_POD=$(kubectl get pod -l app=httpbin,version=v1 -o jsonpat
h={.items..metadata.name})
$ kubectl logs "$V1_POD" -c httpbin
127.0.0.1 - - [07/Mar/2018:19:02:43 +0000] "GET /headers HTTP/1.1" 200
 321 "-" "curl/7.35.0"
```

```
$ export V2_POD=$(kubectl get pod -l app=httpbin,version=v2 -o jsonpat
h={.items..metadata.name})
$ kubectl logs "$V2_POD" -c httpbin
<none>
```

Mirroring traffic to v2

1. Change the route rule to mirror traffic to v2:

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: httpbin
spec:
  hosts:
    - httpbin
  http:
  - route:
    - destination:
        host: httpbin
        subset: v1
      weight: 100
    mirror:
      host: httpbin
      subset: v2
    mirrorPercentage:
      value: 100.0
EOF
```

This route rule sends 100% of the traffic to `v1`. The last stanza specifies that you want to mirror (i.e., also send) 100% of the same traffic to the `httpbin:v2` service. When traffic gets mirrored, the requests are sent to the mirrored service with their Host/Authority headers appended with `-shadow`. For example, `cluster-1` becomes `cluster-1-shadow`.

Also, it is important to note that these requests are mirrored as “fire and forget”, which means that the responses are discarded.

You can use the `value` field under the `mirrorPercentage` field to mirror a fraction of the traffic, instead of mirroring all requests. If this field is absent, all traffic will be mirrored.

2. Send in traffic:

```
$ kubectl exec "${SLEEP_POD}" -c sleep -- curl -sS http://httpbin:8000
/headers
```

Now, you should see access logging for both `v1` and `v2`. The access logs created in `v2` are the mirrored requests that are actually going to `v1`.

```
$ kubectl logs "$V1_POD" -c httpbin
127.0.0.1 - - [07/Mar/2018:19:02:43 +0000] "GET /headers HTTP/1.1" 200
 321 "-" "curl/7.35.0"
127.0.0.1 - - [07/Mar/2018:19:26:44 +0000] "GET /headers HTTP/1.1" 200
 321 "-" "curl/7.35.0"
```

```
$ kubectl logs "$V2_POD" -c httpbin
127.0.0.1 - - [07/Mar/2018:19:26:44 +0000] "GET /headers HTTP/1.1" 200
 361 "-" "curl/7.35.0"
```

Cleaning up

1. Remove the rules:

   ```
   $ kubectl delete virtualservice httpbin
   $ kubectl delete destinationrule httpbin
   ```

2. Shutdown the `httpbin` service and client:

   ```
   $ kubectl delete deploy httpbin-v1 httpbin-v2 sleep
   $ kubectl delete svc httpbin
   ```